A BRANCH AND BOUND ALGORITHM FOR THE DELIVERY TRUCK PROBLEM

Stephen John Balut

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A BRANCH AND BOUND ALGORITHM
FOR
THE DELIVERY TRUCK PROBLEM

by

Stephen John Balut

Thesis Advisor: G. T. Howard

June 1973

Approved for public release; distribution unlimited.

A Branch and Bound Algorithm for the Delivery Truck Problem

by

Stephen John Balut
Lieutenant Commander, United States Navy
Ph.D., Naval Postgraduate School, 1973

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE WITH MAJOR IN MATHEMATICS

from the

ABSTRACT

The delivery truck problem is one in which a truck is loaded with m packages, one package to be delivered to each of m destinations. The amount of fuel consumed by the truck is directly dependent upon the current total weight of the truck, which includes both the weight of the packages and the amount of fuel remaining in the tank. The problem is to determine a sequence in which to deliver all m packages which will minimize total fuel consumption. A branch and bound algorithm for obtaining optimal solutions to the delivery truck problem is presented, along with several sample problems with their solutions. A brief report of computational experience is included.

TABLE OF CONTENTS

I. INTRODUCTION

The delivery truck problem is one in which a truck is loaded with m packages, one package to be delivered to each of m destinations. The amount of fuel consumed by the truck is directly dependent upon the current total weight of the truck, which includes both the weight of the packages and the amount of fuel remaining in the tank. The problem is to determine a sequence in which to deliver all m packages which will minimize total fuel consumption.

This problem is closely related to the traveling salesman problem. It differs in that the route is not closed and that fuel consumed on each leg of the journey is not only dependent upon the length of the leg, but also the current weight of the truck.

This thesis is the presentation of an algorithm for obtaining optimal solutions to the delivery truck problem. The methodology of branch and bound is used to implicitly enumerate all possible solutions. Several example problems with solutions are presented followed by a brief report of computational experience for problems with m ranging from five to twenty.

II. RELATED RESEARCH

The delivery truck problem is clearly a problem in the area of discrete mathematical optimization. Reference 8 is a current summary of results applicable to such problems.

Current research on the ordinary traveling salesman problem is summarized in Ref. 2, in which, after an analysis of computational results, it is recommended that dynamic programming [Ref. 5] be used for problems with thirteen cities or less, and that branch and bound [Ref. 9] be used for the rest.

An algorithm for obtaining optimal solutions to the modified traveling salesman problem in which no return to the starting city is required is presented in Ref. 7.

In an extension of the traveling salesman problem called "The Delivery Problem" [Ref. 4] more than one salesman is available to visit the predetermined set of cities.

Another extension of the traveling salesman problem, sometimes referred to as the "Milk Bottle Problem", [Ref. 4] is that in which the milkman, (salesman), is to deliver one or more bottles of milk to a known set of houses, (cities). He must walk and wishes to minimize the amount of work expended carrying the milk bottles. The problem is very closely related to the delivery truck problem, the difference being that no account is taken for fuel

used along the way. A branch and bound algorithm for the milk bottle problem is presented in Ref. 3. This algorithm is a direct extension of the work of Little, et al., [Ref. 6].

The delivery truck problem can be formulated as an investigation problem. Current results in Investigation Theory are contained in Ref. 1. The algorithm presented in this thesis uses an approach similar to that of the branch and bound algorithm presented in Ref. 1.

III. MATHEMATICAL DESCRIPTION

The following notation will be used. Let

(d_{ij}) = a matrix of distances between destination i and destination j, $i = 1,2,\ldots,n$, $j = 2,3,\ldots,n$

π = $(\pi_1, \pi_2, \ldots, \pi_n)$ be a permutation of the integers $1,2,\ldots,n$ representing an order in which destinations $1,2,\ldots,n$ are visited.

f_π = the amount of fuel required if path π is followed.

f_{π_i} = the amount of fuel required if destinations $\pi_1, \pi_2, \pi_3, \ldots, \pi_i$ are visited in that order. Clearly $f_\pi = f_{\pi_n}$.

$F_{\pi_j}(i, f_{\pi_i})$ = fuel required to travel from destination π_i to destination π_j given that i destinations have already been visited and f_{π_i} fuel has been consumed.

Let $f_{\pi_o} = 0$. Then

$$f_{\pi_i} = \sum_{j=0}^{j=i-1} F_{\pi_{j+1}}(j, f_{\pi_j}).$$

The delivery truck problem is to select that path π which minimizes f_π.

The algorithm contained herein assumes it is known which destination will be the first visited, and the problem starts with the truck initially located there. If the first destination is not known, the truck can be assumed to be located at a dummy location added to the given set of destinations, with travel allowed to any valid destination at zero fuel consumption.

Given that the truck is initially located at destination one, there are (n-1)! possible solutions.

In different variations of the delivery truck problem, different fuel consumption functions F are appropriate. In this thesis, F is taken to have the following form:

$$F_{\pi_j}(i, f_{\pi_i}) = c_1(c_2 - ic_3^k - f_{\pi_i})d_{\pi_i\pi_j}$$

where

c_1 = an arbitrarily chosen constant

c_2 = the fully loaded weight of the truck

c_3^k = the weight of package k, $k = 1,\ldots,n$.

Note that if $f_{\pi_i} = 0$ for all i, the problem reduces to the milk bottle problem of Ref. 3. If, in addition, $c_3^i = 0$, $i = 1,\ldots,n$, the problem reduces to the modified traveling salesman problem in which no return to the starting city is required.

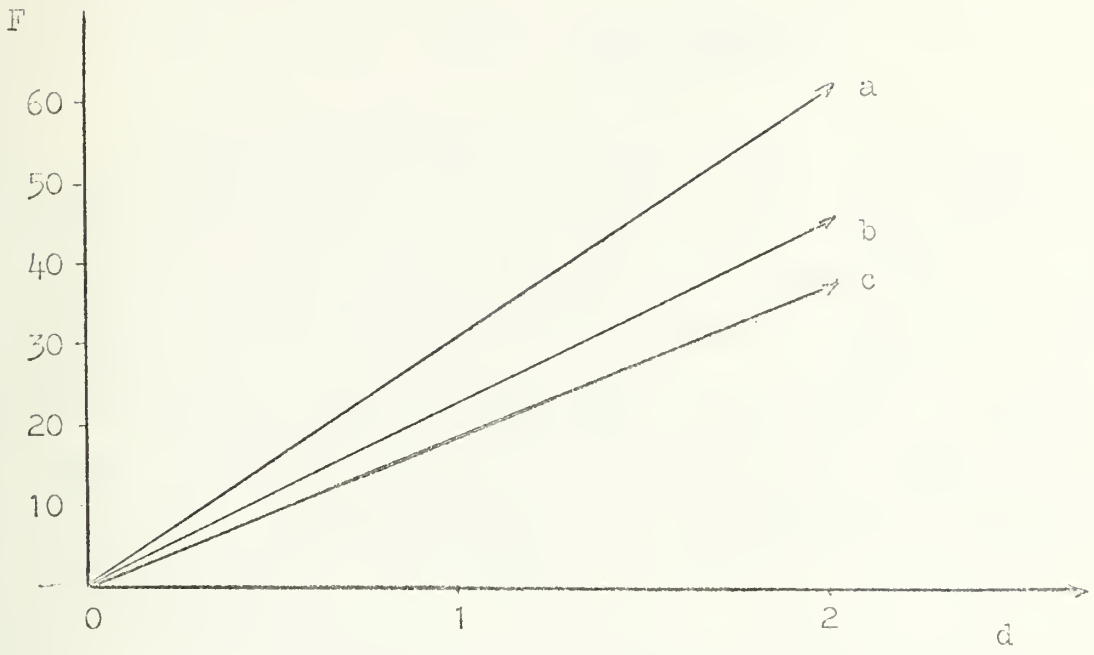


Figure 3.1

For the experimental testing discussed in Section IV(E) the constants chosen were $c_1 = .004$, $c_2 = 8000$ and $c_3^i = 270$, for $i = i,2,\ldots,n$. The resulting function is used to illustrate variation in fuel consumption due to truck loading as shown in Figure 3.1. Line a represents consumption with i = 0 and f = 0, line b for i = 5, f = 0, and line c for i = 5 and f = 500.

IV. THE ALGORITHM

This section begins by describing how the branch and bound method is applied to the delivery truck problem, followed by a general statement of the algorithm. (A flow diagram of the algorithm is contained in Appendix A and the computer program immediately afterwards.) Several sample problems are used to illustrate the output of the algorithm. A brief summary of computational experience is presented.

A. APPLICATION OF BRANCH AND BOUND TO THE DELIVERY TRUCK PROBLEM

The branch and bound method consists of selectively partitioning the set of all feasible solutions and computing bounds on the objective function for each element of the partition. This process is continued until one element of the partition containing a single solution is obtained for which the associated bound is at least as good as that of any other element.

A branching tree is initiated and extended such that each node at the end of a branch of the tree corresponds to a set of solutions. The set of all such nodes specifies a partition of the solution space. The bounds associated with each element of the partition, and therefore with each node at the end of a branch of the tree, represent a

lower bound on the amount of fuel used if any solution contained in that element of the partition is followed.

Associated with every node of the tree is a destination. The initial node, representing all solutions, corresponds to destination number one, the initial location of the truck. Branching corresponds to selecting a node from which to branch and in specifying a destination for use in extending the path specified by that branch. The branch corresponding to node one naturally is selected initially for extension. This branch is extended to the right and the left by adding two new nodes to the tree. See Figure 4.1. The node labeled k represents all solutions in

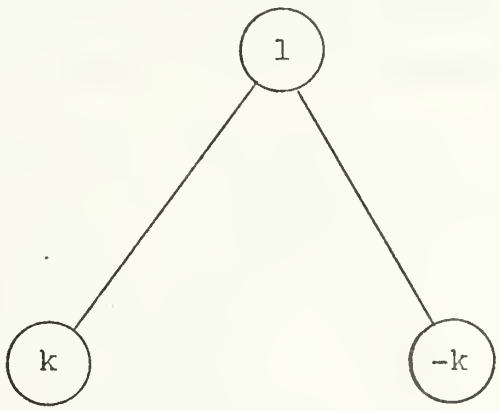


Figure 4.1 Initial branching

which the truck starts at destination one and proceeds first to destination k. The node labeled -k represents all solutions in which the truck does not go to destination k first.

Associated with each node at the end of a branch of the tree is a path, starting with destination one, and including each destination number corresponding to a positively labeled node in the order in which they appear in the branch.

When a branch of the tree is selected for extension, the path specified by that branch is called the current path.

The bound on each ending node with a positive label applies to all solutions starting with the path specified by that node. The bound on those with negative labels corresponds to all solutions starting with the path specified which do not include as their next element, all negative labels of nodes at the end of that branch. In illustration of this, consider Figure 4.2. The bound on node labeled 5

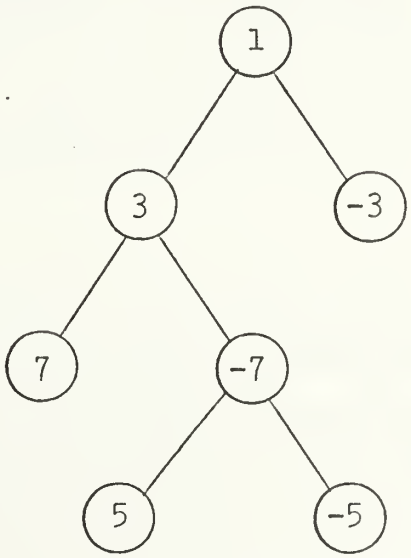


Figure 4.2 Sample tree

applies to all solutions starting with elements 1, 3, 5, in that order. The bound on the node labeled -5 applies to all solutions starting with elements 1, 3, and not having 7 or 5 as the next element.

After a branch is selected for extension, it is necessary to determine which destinations are eligible for use in extending the current path. This is done by determining all destinations not in the current path and not prohibited due to negatively labeled nodes at the end of the branch.

The order of an element of the partition reduces to one when there are no destinations eligible for use in extending the branch corresponding to that element. When such an element is located with a bound less than or equal to the bound on all others, the path corresponding to the sole solution contained in that element is optimal.

B. GENERAL STATEMENT OF THE ALGORITHM

Specific methods of branching and bounding are ignored temporarily while the general structure of the algorithm is presented.

Algorithm:

1. Create the initial node corresponding to destination number one. Compute a bound for this node. Go to 2.
2. Select a branch to extend. This branch determines the current path. Go to 3.
3. Determine the set E of destinations eligible for use in extending the current path. Put into E all destinations

not in the current path. Remove from E all indices corresponding to negatively labeled nodes at the end of the branch being extended. If $|E| = 0$, stop. If not, go to 4.

4. Select from E a destination k for use in extending the current path. Extend the branch to the left and right creating two new nodes labeled k and -k. Go to 5.

5. Compute bounds for branches ending with nodes labeled k and -k. Go to 2.

Upon termination the current path is optimal.

C. BRANCHING AND BOUNDING RULES

1. Branching

Branching is a two part operation, consisting first of selecting a branch to extend, and next of selecting a destination for use in extending that branch. It has proved to be advantageous in branch and bound to use as the first of these steps the selection of that branch whose associated bound is smallest and specifying tie breaking rules in the event they occur. This rule is used in the algorithm.

The second part of the branching operation selects a destination for use in extending the current path. The algorithm selects that eligible destination which is closest to the last element in the current path, or, in other words, the unvisited destination closest to the truck's current position.

2. Bounding

The bound associated with each branch specifies a lower bound on the amount of fuel used if the path specified by that branch is followed. In illustration of the method used, suppose branching is being carried out from a node labeled j to destination k, and that there are r destinations in the current path. The exact amount of fuel required to follow the current path is f_j, and the fuel required to travel from destination j to destination k is $F_k(r, f_j)$. The lower bound on the node labeled k is

$$f_j + F_k(r, f_j) + L$$

where L is a lower bound on the fuel required to visit the remaining n-r-1 destinations.

L is obtained by considering the upper triangle of the reduced matrix (d_{ij}) where i and j are not destinations in the current path, selecting the smallest element in row k and the remaining n-r-2 smallest elements, ordering these distances in non-decreasing order, and computing the amount of fuel required to traverse these distances in the prescribed sequence.

Only the upper triangle need be considered since (d_{ij}) is symmetric and travel between destination i and destination j is carried out at most once.

It can be seen that L is a valid lower bound by noting that after leaving destination k, the truck must

traverse an additional n-r-1 legs in its required journey. Clearly the shortest distance it will be required to haul n-r-1 packages is the smallest element in row k of the reduced matrix. A lower bound on the next leg is clearly the fuel required to haul n-r-2 packages the smallest distance in the reduced matrix, excluding the element used for the first leg. This procedure is continued until all n-1 legs of the journey are accounted for.

When the bound is computed for the node labeled -k, the procedure is the same except that selection of a distance for the first additional leg after leaving destination j is not restricted to come from row k of the reduced matrix.

D. PROOF OF OPTIMALITY

Let S be the set of all solutions to the problem, i.e.,

$$S = \{\pi | \pi \text{ is a permutation of integers } 1,2,\ldots,n\} ,$$

and let P be a partition of S such that

$$P = \{P_1, P_2, \ldots, P_m\} .$$

Let b_i be a lower bound on all solutions contained in P_i, i.e.,

$$b_i \leq f_\pi \quad \text{for all } \pi \text{ in } P_i .$$

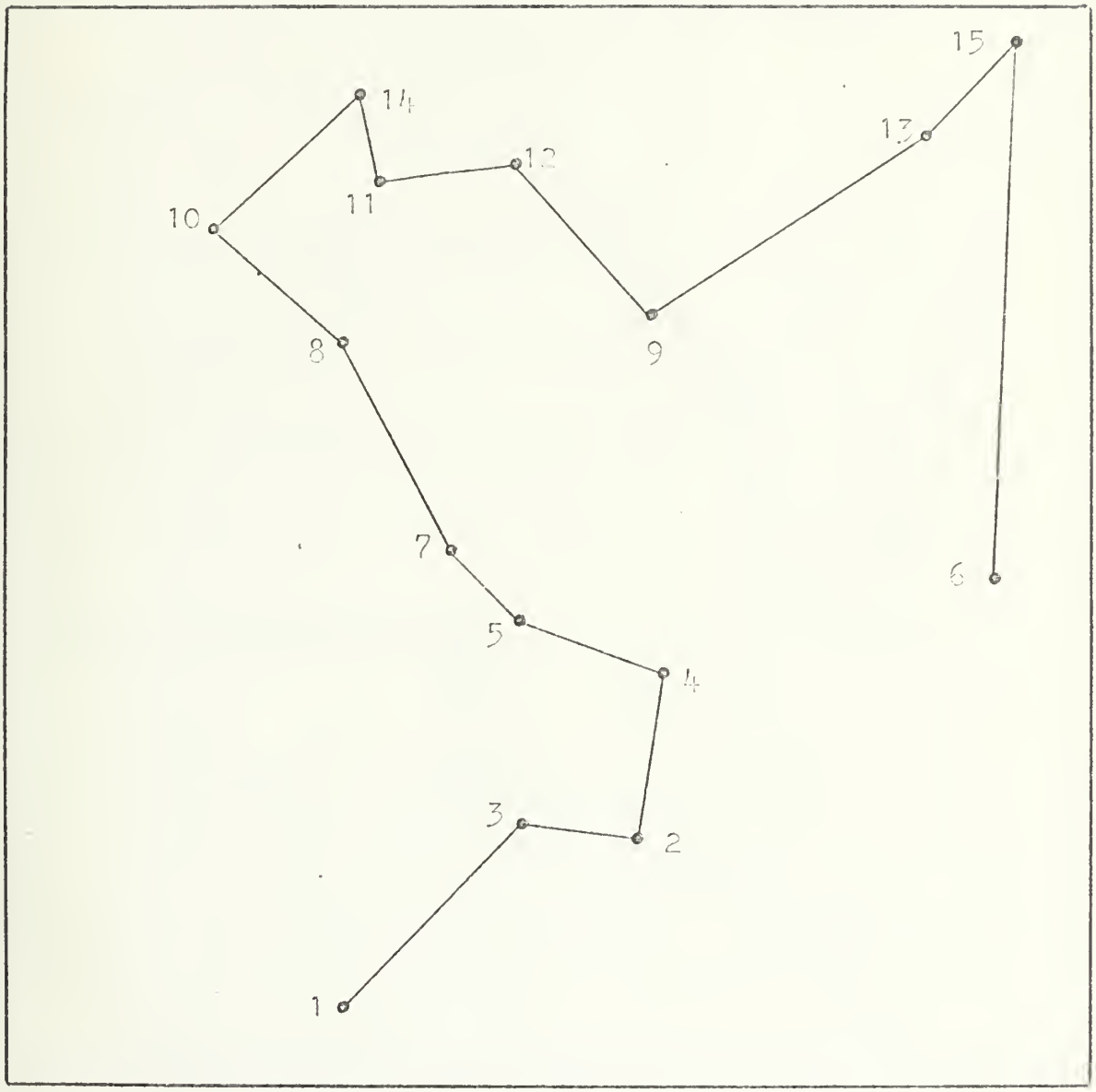
In Step 2 of the algorithm a branch of the tree is selected for extension. Selecting a branch for extension corresponds to selecting an element of the current partition for refinement. If element P_j is selected and $|P_j| = 1$, no further refinement is possible, and in Step 3, $|E| = 0$. At this point all n destinations are contained in the current path, and bound b_j is no longer a lower bound, but exact. Note that the method of selection used for P_j, as described in Section IV(C)1, is to select partition element j such that

$$b_j = \min b_k \quad \text{for all } P_k \text{ in } P.$$

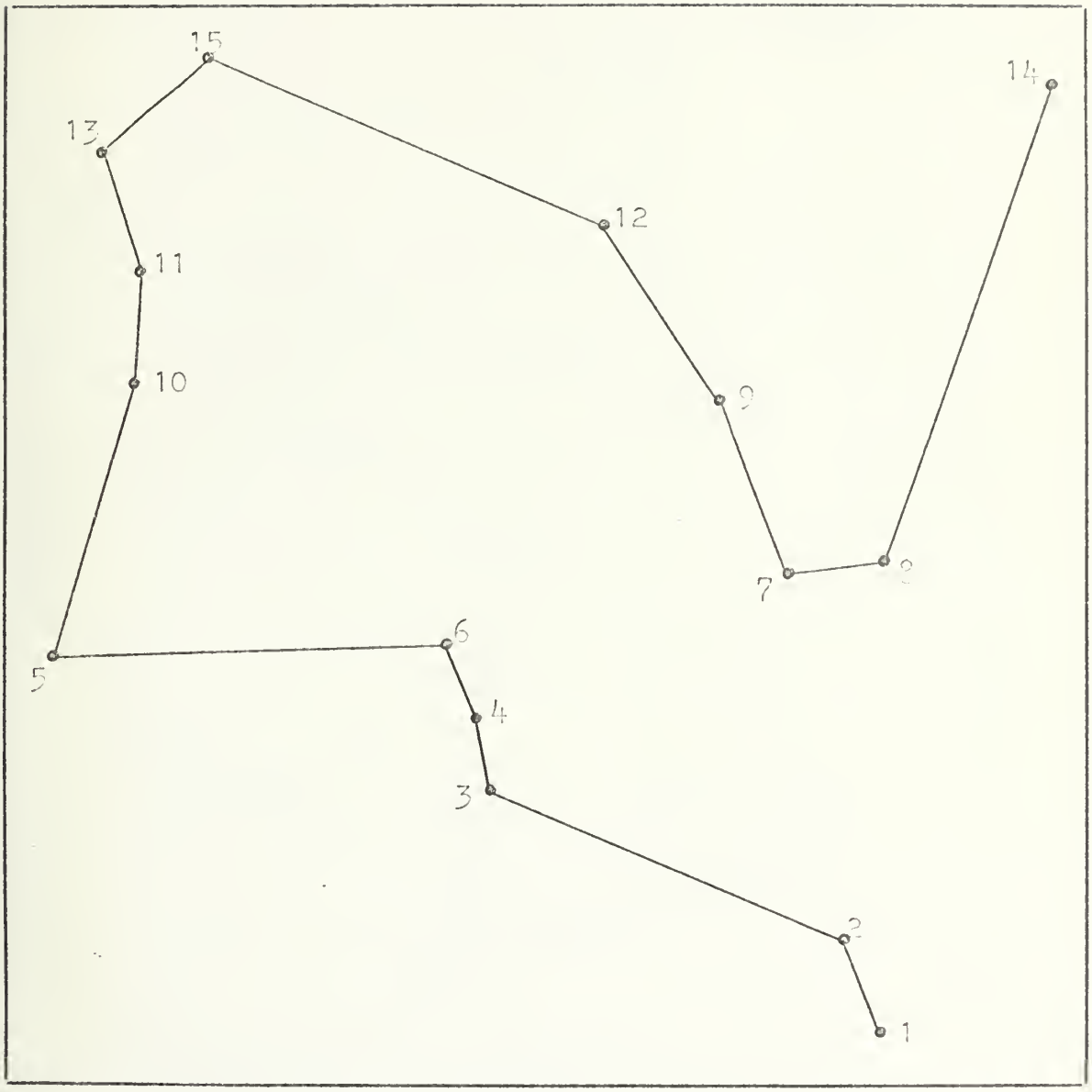
Hence, P_j has an exact bound which is less than or equal to the lower bounds associated with all elements of P. The single solution π in P_j is thus a solution which minimizes f_π.

E. SAMPLE PROBLEMS AND SOLUTIONS

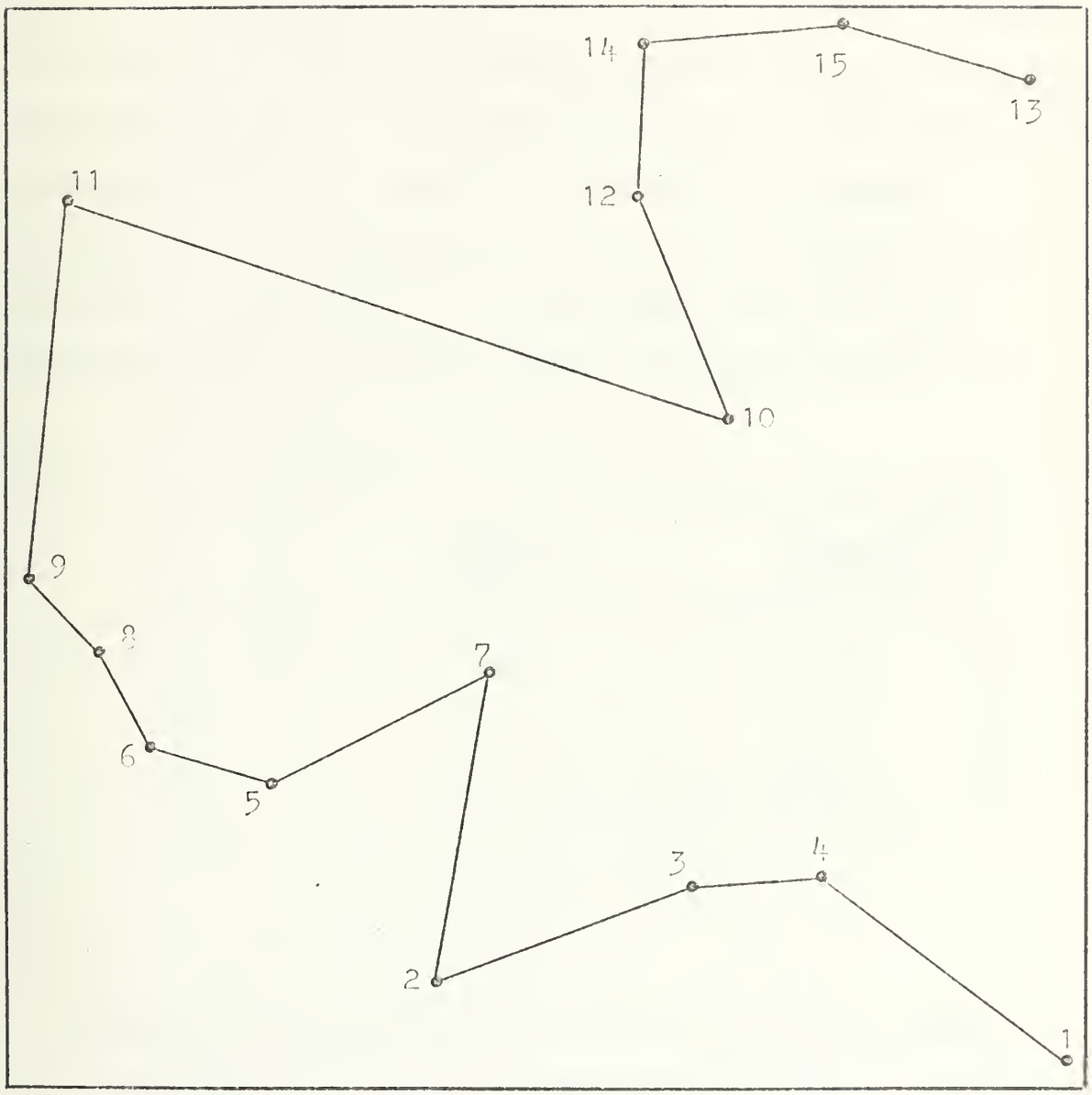
The following are sample problems with optimal solutions. The x and y coordinates of the destinations were selected from a uniform distribution over the interval (0, 10). Coordinates were adjusted slightly when necessary to insure that each pair of destinations was at least a unit distance apart. The truck is assumed to be initially located at destination one. The path shown is the one which visits each of the remaining destinations and uses the minimum amount of fuel.



Sample Problem 1



Sample Problem 2



Sample Problem 3

F. COMPUTATIONAL EXPERIENCE

Computational experience with the algorithm is limited, but an investigation was made into the way the time required to solve an n destination problem varies with n. Seventeen sample problems were developed as described in the preceeding section; five with n equal to five, six with n equal to ten, and six with n equal to fifteen. The average times required to solve each set of problems, along with the average number of bounds computed, is shown in Table 4.1.

n	average time (sec.)	average # of bounds
5	.38	5.1
10	1.6	145.0
15	99.7	2525.0

Table 4.1

Although these results are sparse, this data suggests that time required increases exponentially with n, as seen in Figure 4.9. The circles show the data points of Table 4.1. The dots show times required to solve sample problems having n equal to 20.

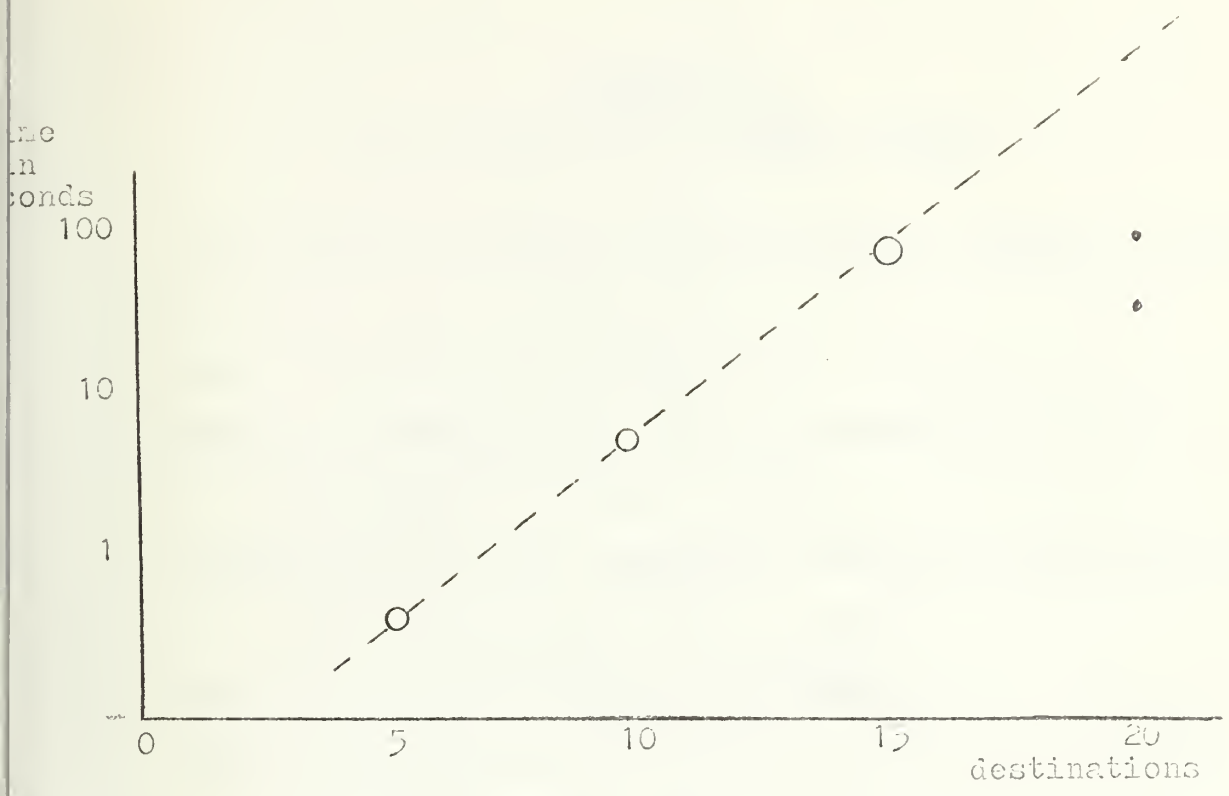


Figure 4.9

APPENDIX A

FLOW DIAGRAM OF COMPUTER PROGRAM

The following notation, not previously defined, is used:

LN = the last node created

HN(I) = destination number associated with node I, for I = 1, 2, ..., LN

PRED(I) = the node immediately node I in the branching tree, for I = 2, ..., LN

BFN = node selected for extension

BTH = destination selected for use in extending the current path

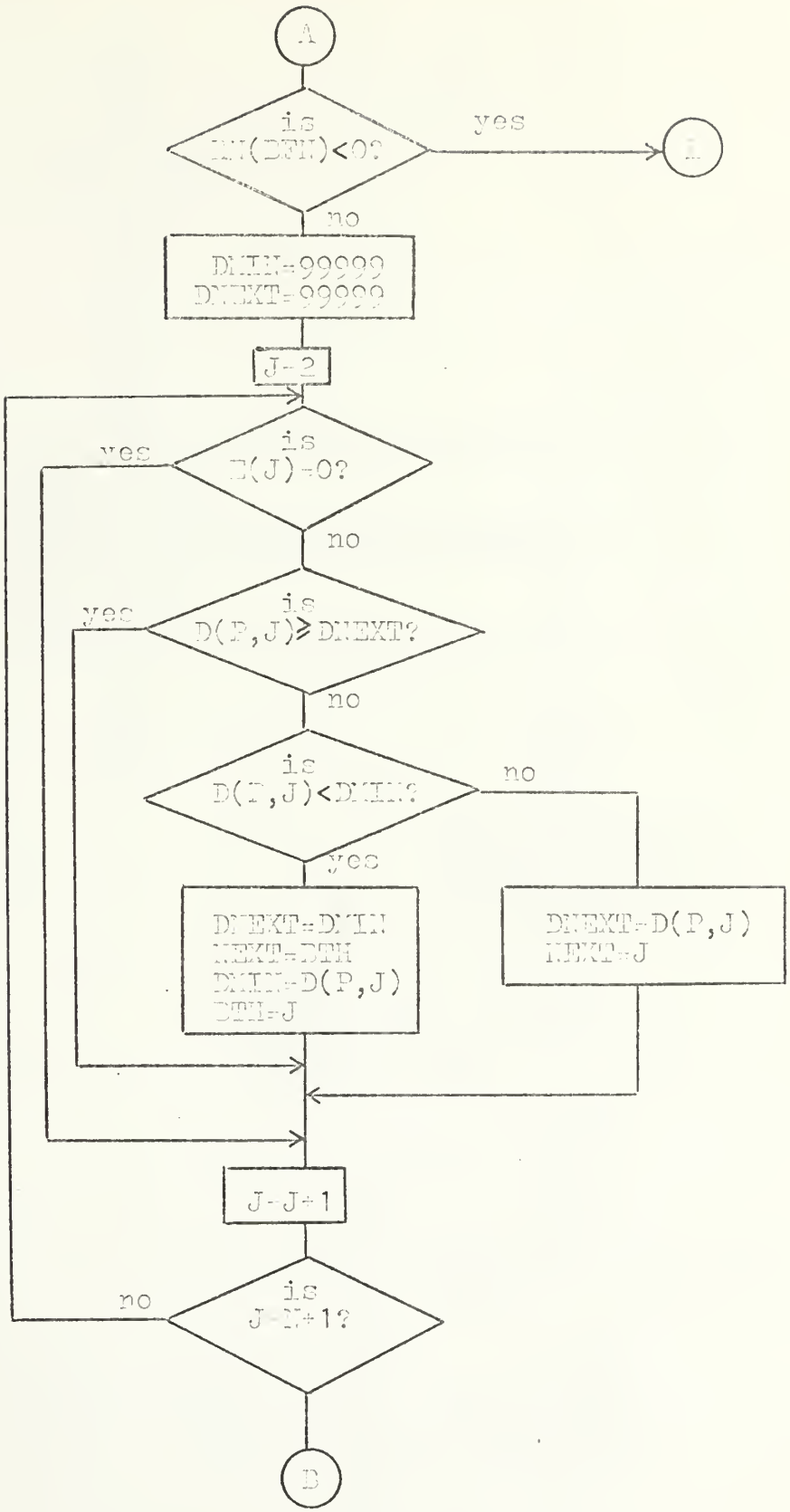
BB(I) = bound on node I

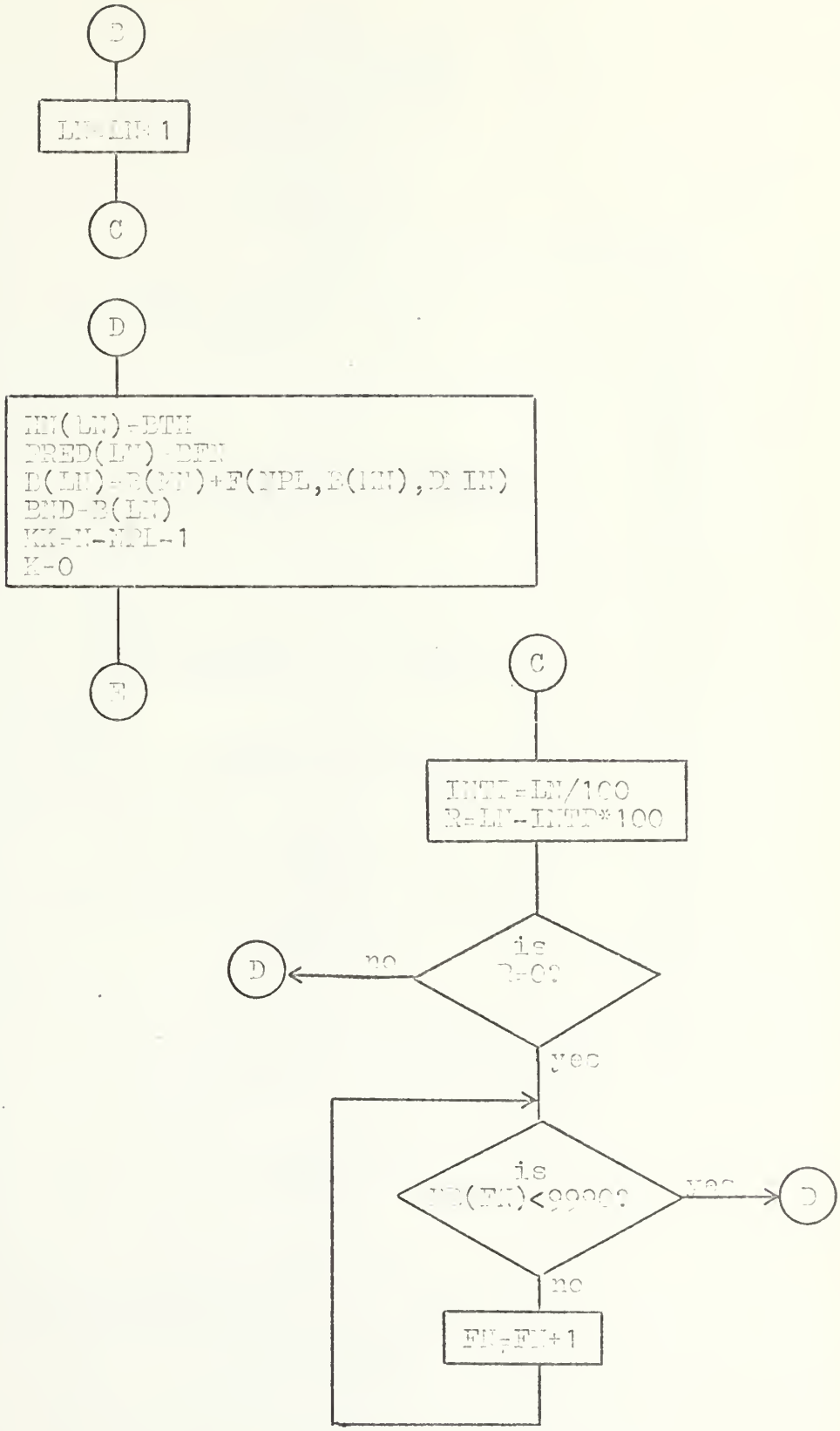
B(I) = BB(I) is node I is at the end of a branch in the tree, and equals M otherwise, where M is large.

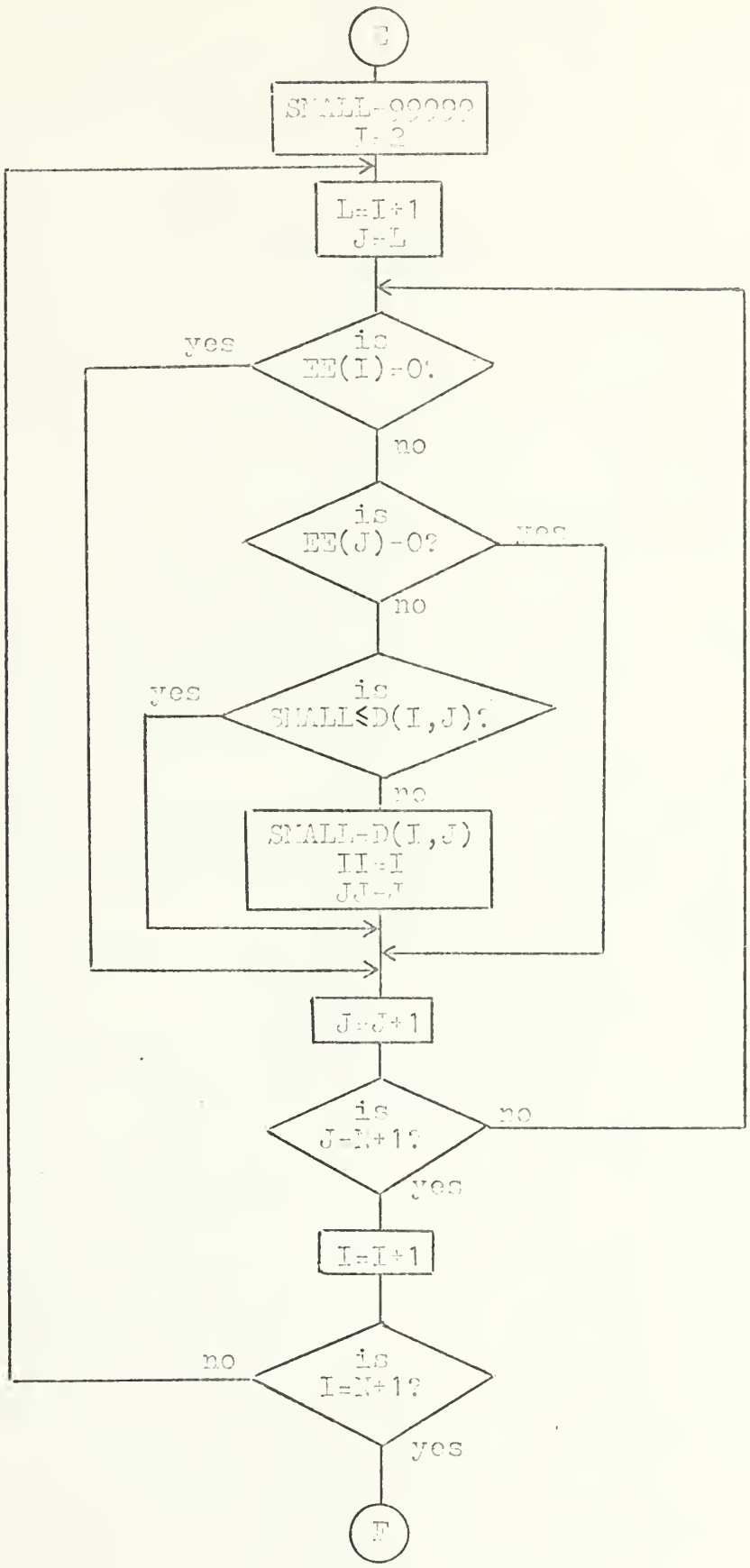
NPL = number of packages currently delivered

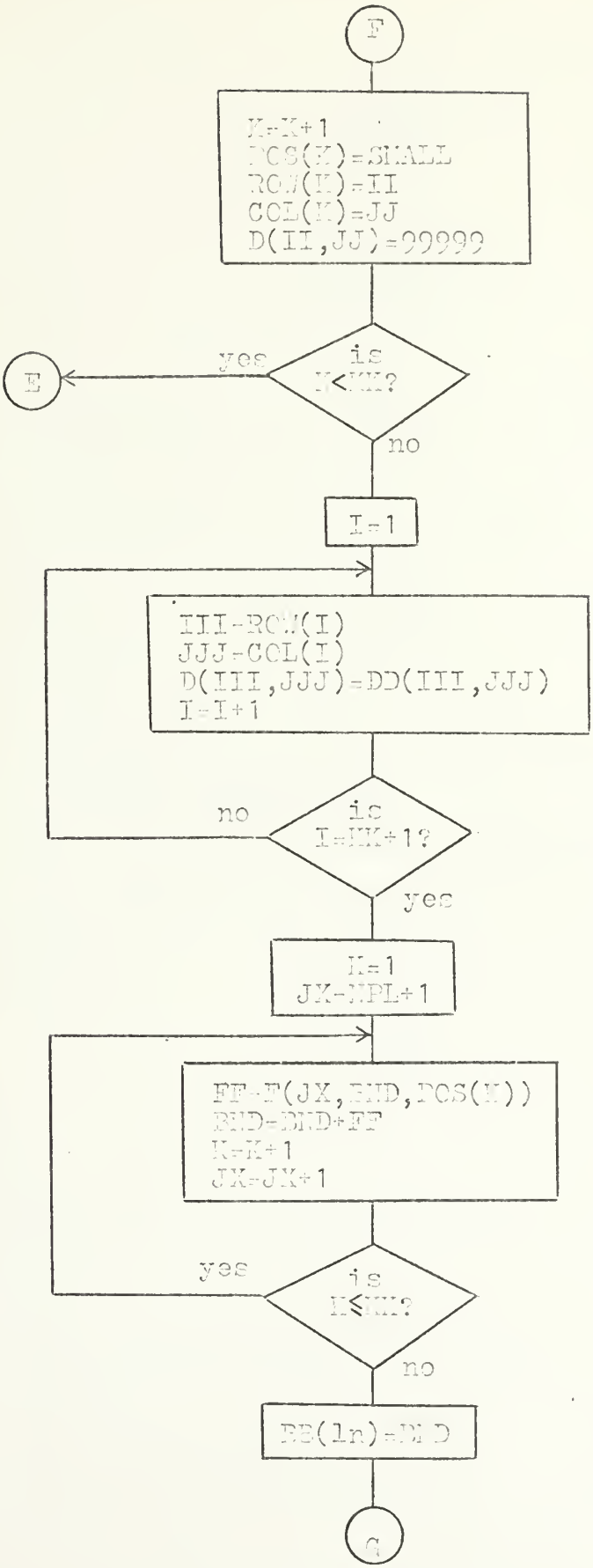
MN = destination at which the truck is currently located

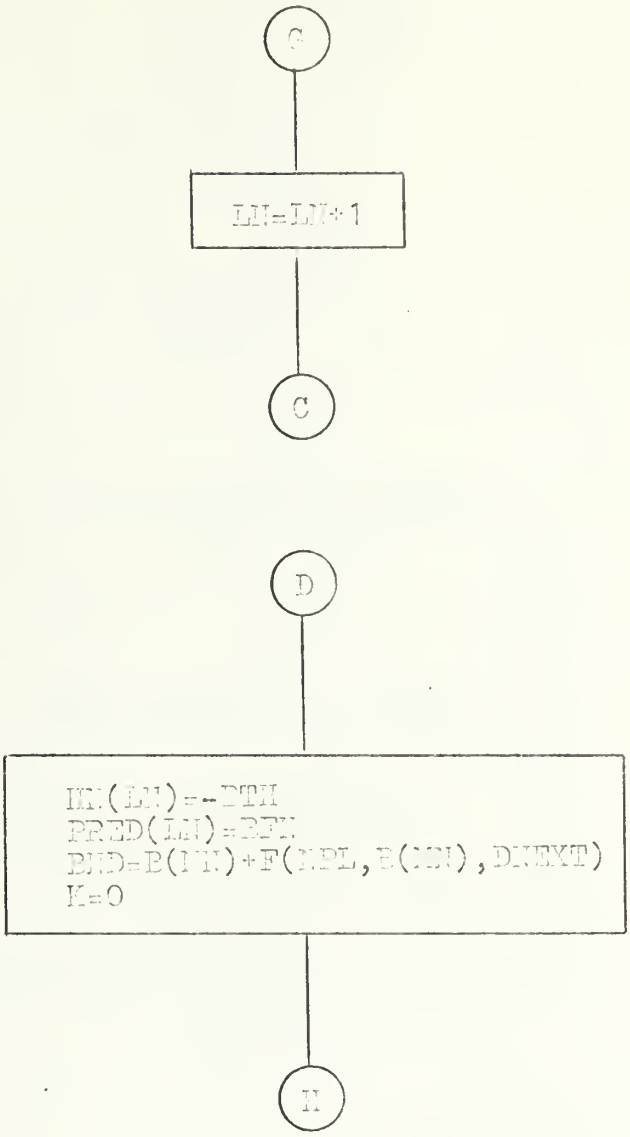
F(X,Y,Z) = fuel required to travel distance Z if X packages have currently been delivered and Y fuel consumed doing so.

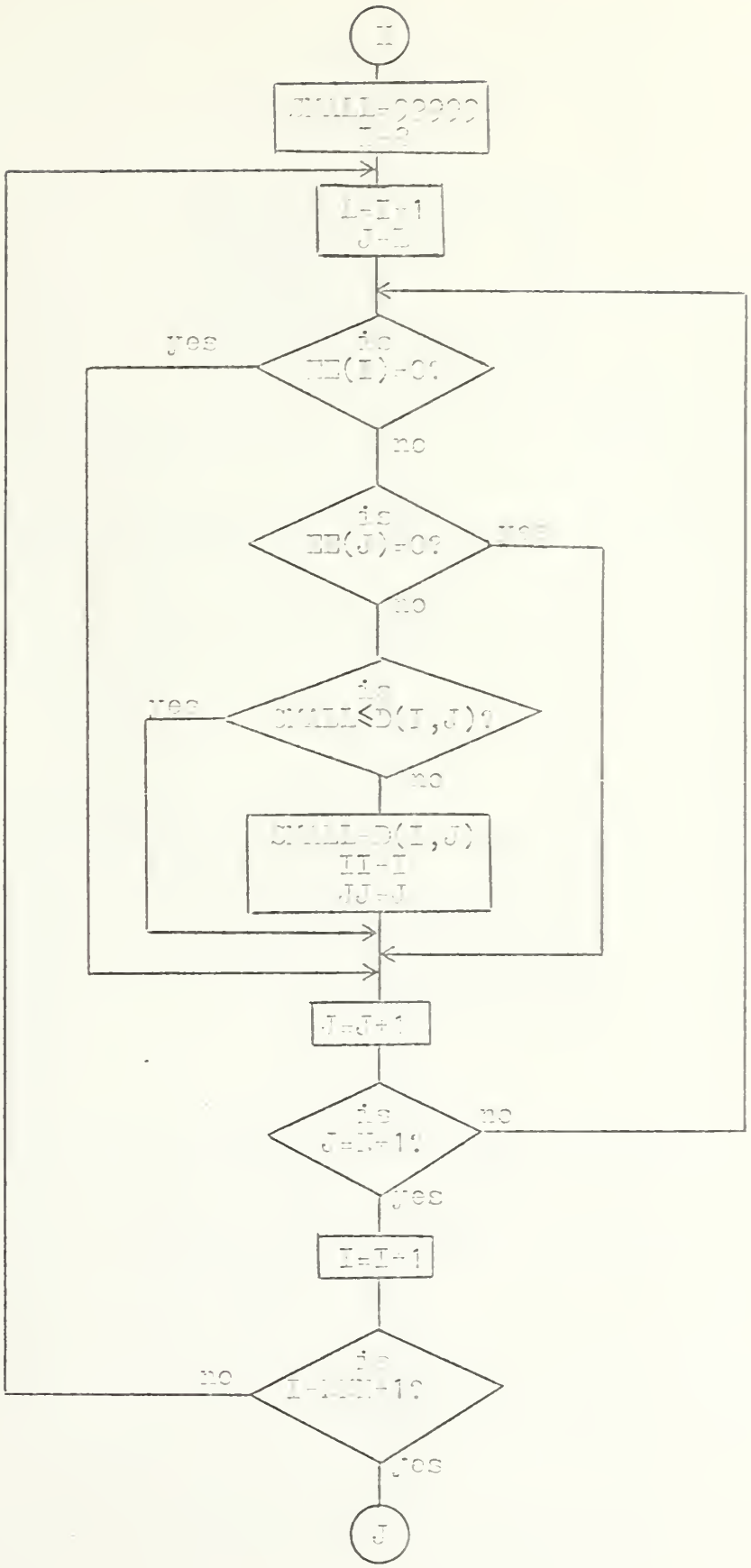

A
is
MN(DTH)<0?
yes
1
no
DMIN=99999
DNEXT=99999
J=2
is
E(J)=0?
yes
no
is
D(P,J)≥DNEXT?
yes
no
is
D(P,J)<DMIN?
no
yes
DNEXT=DMIN
NEXT=DTH
DMIN=D(P,J)
DTH=J
DNEXT=D(P,J)
NEXT=J
J=J+1
is
J=N+1?
no
B

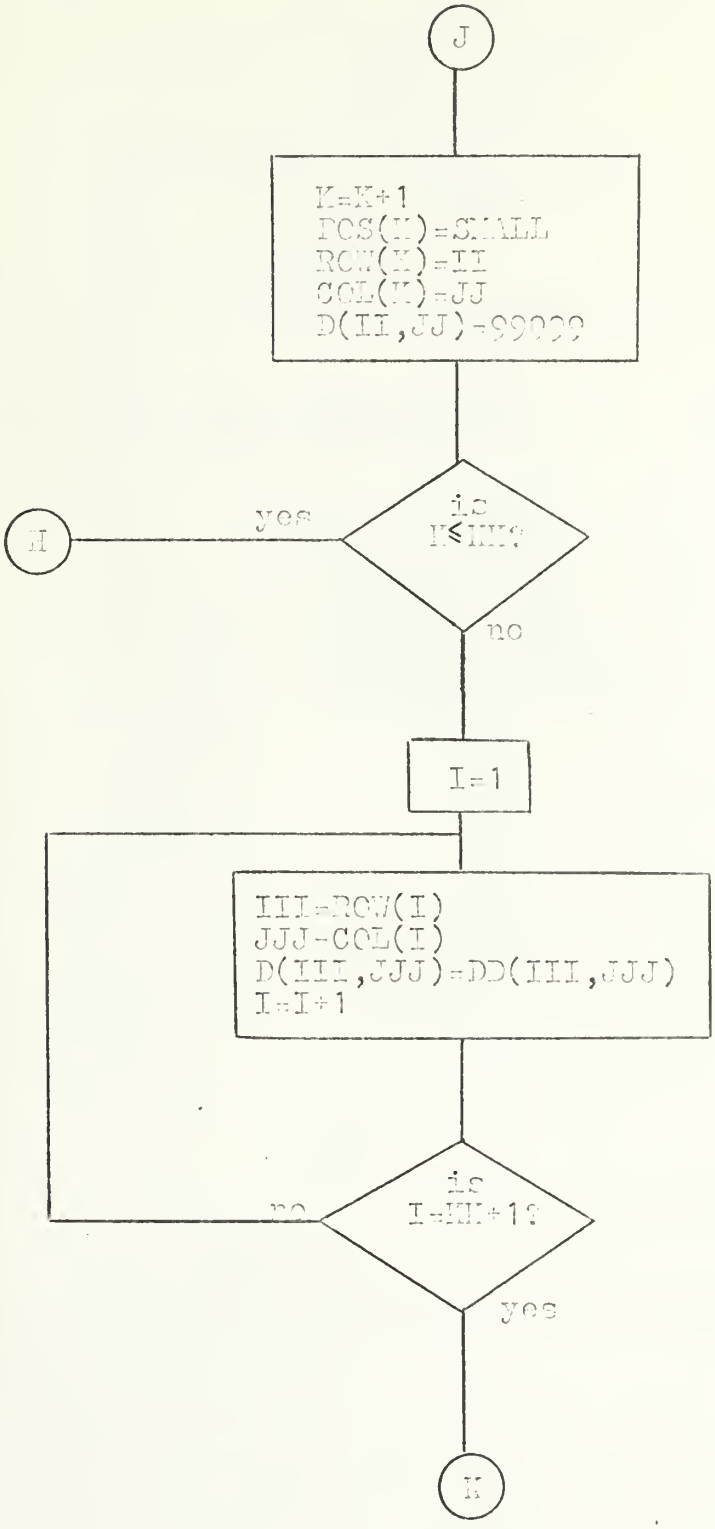

D
LN=LN+1
C
D
INT(LN)=BTH
PRED(LN)=BTH
B(LN)=B(NT)+F(NPL,B(IN),D IN)
BND=B(LN)
KK=N-NPL-1
K=0
E
C
INTT=LN/100
R=LN-INTT*100
is R=0?
no
D
yes
is NR(EN)<9999?
yes
D
no
EN=EN+1

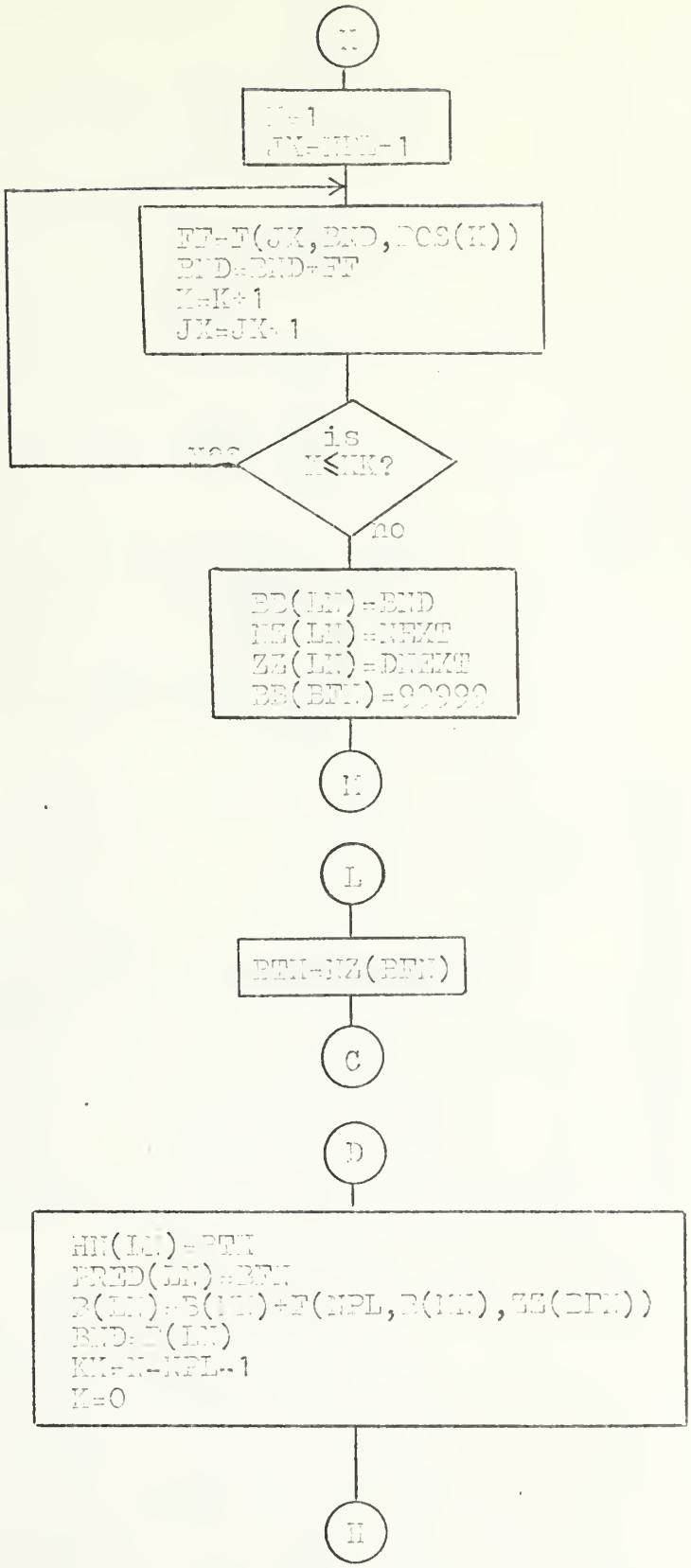

C
SMALL=99999
I=2
L=I+1
J=L
is EE(I)=0?
yes
no
is EE(J)=0?
yes
no
is SMALL≤D(I,J)?
yes
no
SMALL=D(I,J)
II=I
JJ=J
J=J+1
is J=N+1?
no
yes
I=I+1
is I=N+1?
no
yes
F

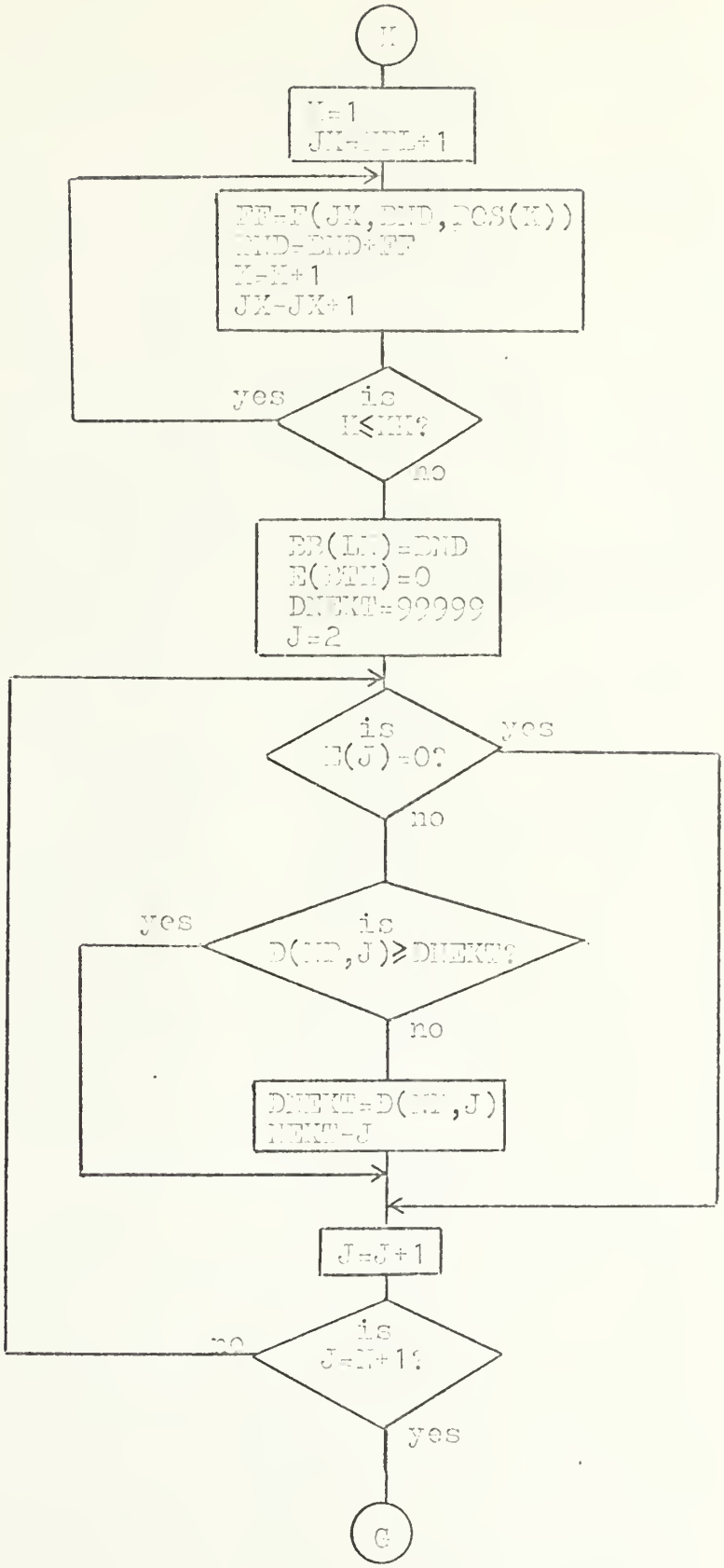

F
K=K+1
POS(K)=SMALL
ROW(K)=II
COL(K)=JJ
D(II,JJ)=99999
yes
is
K<KK?
E
no
I=1
III=ROW(I)
JJJ=COL(I)
D(III,JJJ)=DD(III,JJJ)
I=I+1
no
is
I=KK+1?
yes
K=1
JK=NPL+1
FF=F(JX,END,POS(K))
END=END+FF
K=K+1
JX=JX+1
yes
is
K≤KK?
no
BB(ln)=END
G

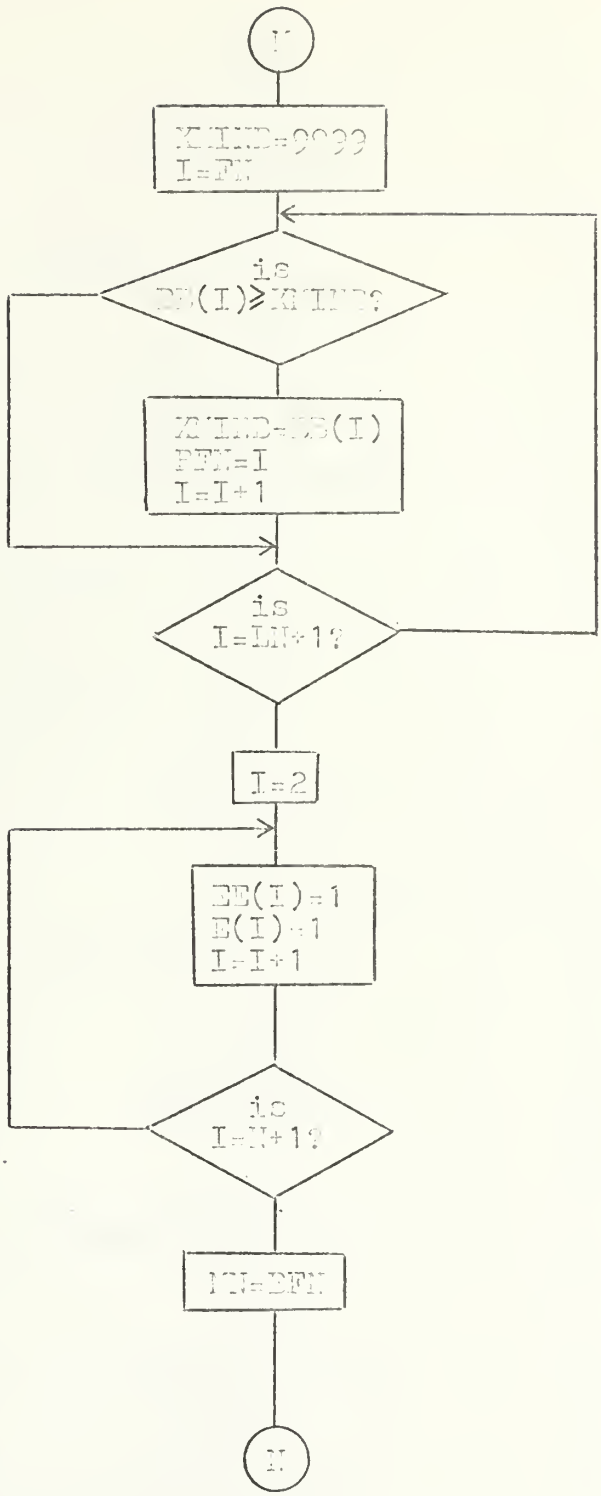

C
LN=LN+1
C
D
IR(LN)=-DTH
PRED(LN)=BTL
BND=B(IR)+F(IPL,B(IBN),DNEXT)
K=0
H

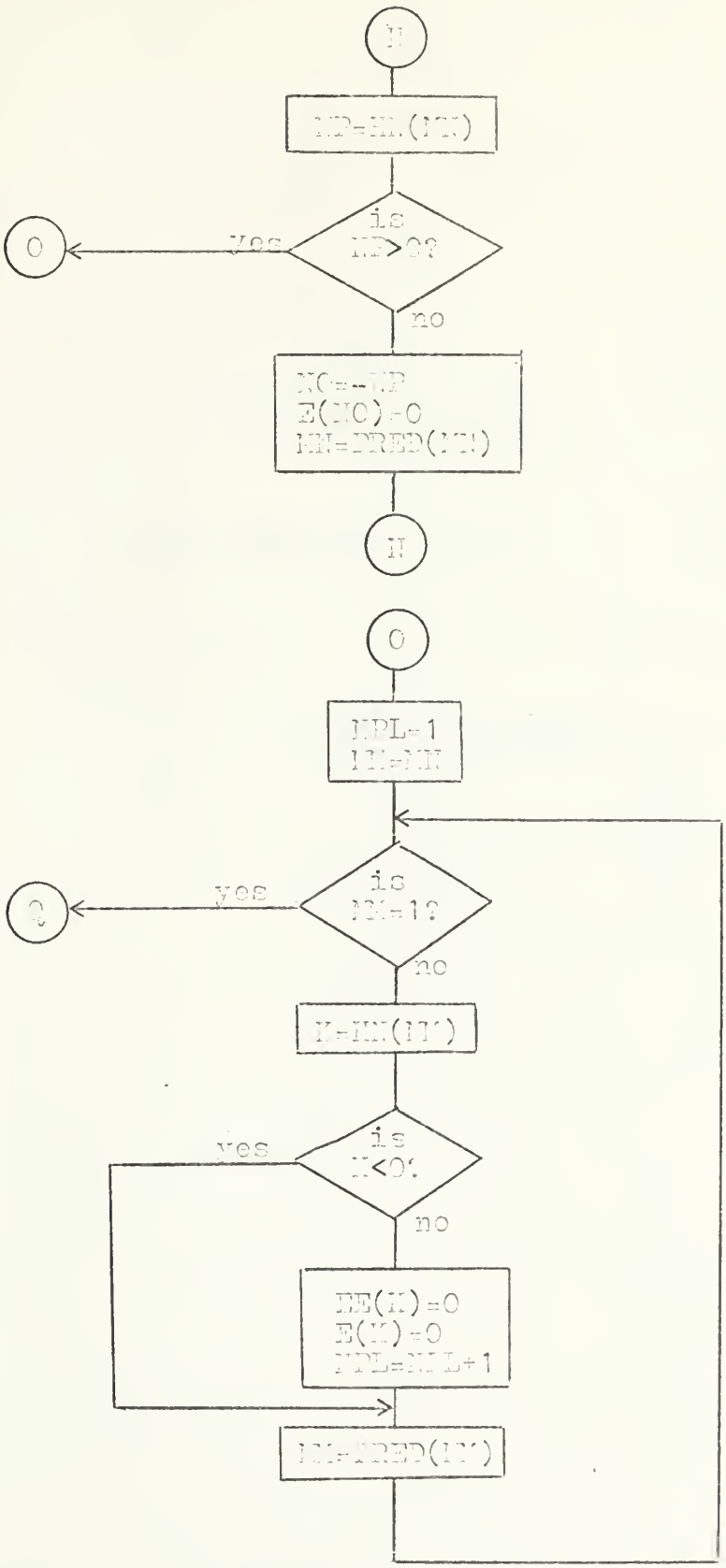

I
I=I+1
J=I
is
II(I)=0?
yes
no
is
JJ(J)=0?
yes
no
is
SMALL≤D(I,J)?
yes
no
SMALL=D(I,J)
II=I
JJ=J
J=J+1
is
J=N+1?
no
yes
I=I+1
is
I=N+1?
no
yes
J

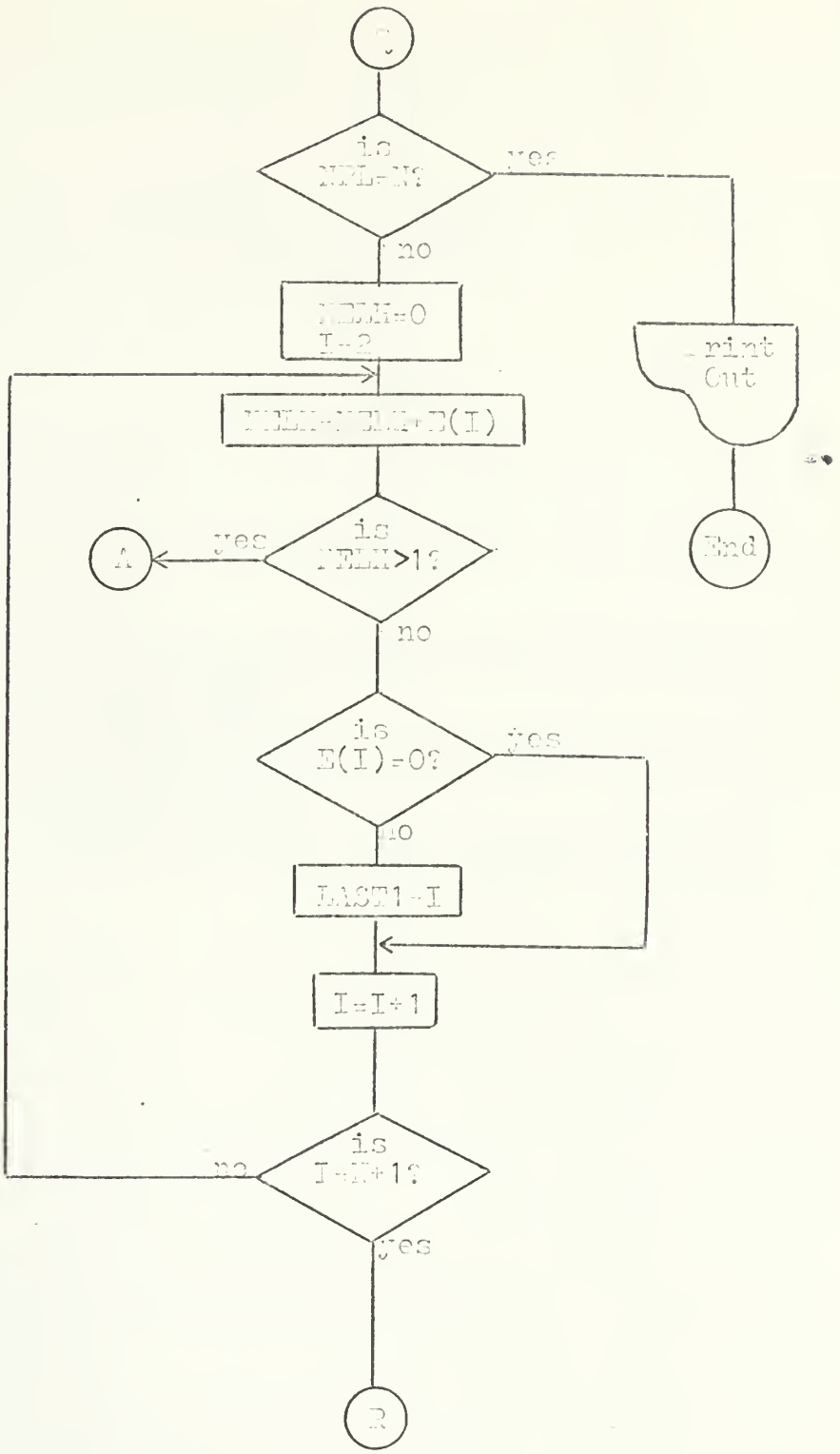

J
K=K+1
POS(K)=SMALL
ROW(K)=II
COL(K)=JJ
D(II,JJ)=99999
is K≤KK?
yes
H
no
I=1
III=ROW(I)
JJJ=COL(I)
D(III,JJJ)=DD(III,JJJ)
I=I+1
is I=KK+1?
no
yes
K

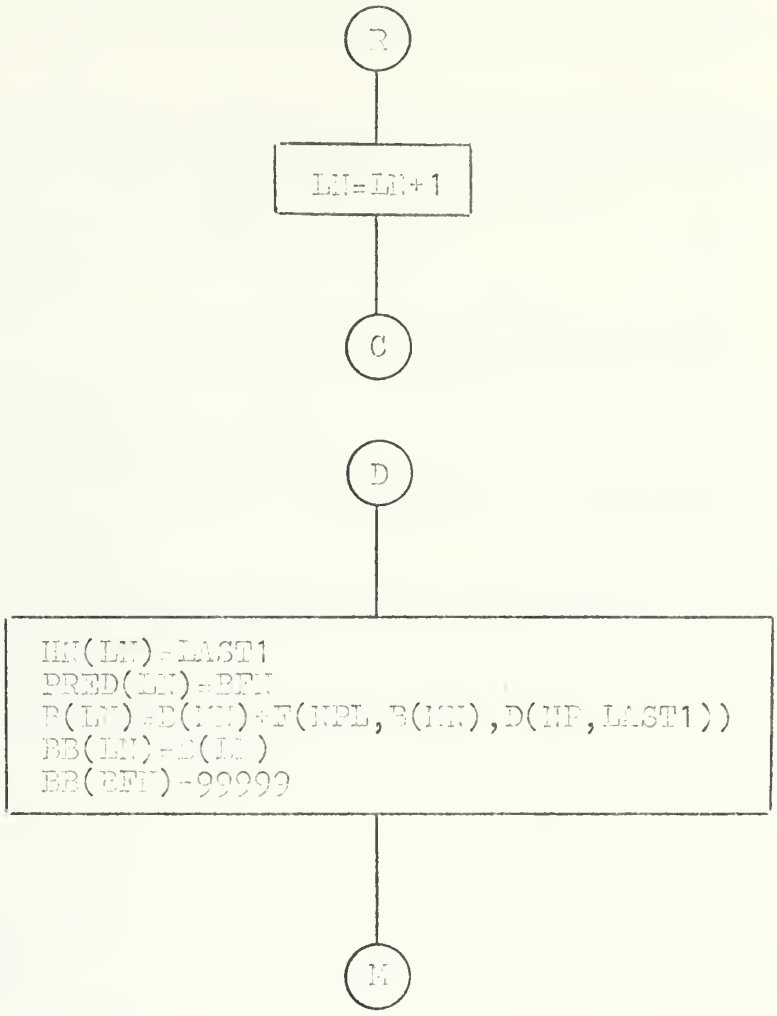

K
K=1
JK=IND-1
FT=F(JK,BND,POS(K))
BND=BND+FT
K=K+1
JK=JK+1
is K≤NK?
yes
no
BD(LN)=BND
NE(LN)=NEXT
ZZ(LN)=DNEXT
BB(BTN)=99999
K
L
BTN=NZ(BTN)
C
D
HN(LN)=BTN
PRED(LN)=BTN
P(LN)=B(NL)+F(NPL,P(NK),ZZ(BTN))
BND=P(LN)
NK=N-NPL-1
K=0
H


K
K=1
JK=KTL+1
FF=F(JK,END,POS(K))
END=END+FF
K=K+1
JK=JK+1
yes
is
K≤KH?
no
BR(LN)=END
E(BTH)=0
DNEXT=99999
J=2
is
E(J)=0?
yes
no
yes
is
D(NP,J)≥DNEXT?
no
DNEXT=D(NP,J)
NEXT=J
J=J+1
is
J=N+1?
no
yes
G


I
XMIND=9999
I=FN
is
DD(I)≥XMIND?
XMIND=DD(I)
FN=I
I=I+1
is
I=LN+1?
I=2
EE(I)=1
E(I)=1
I=I+1
is
I=N+1?
MN=FN
N


H
MP=MM(MM)
is MP>0?
yes
0
no
MO=-MP
E(MO)=0
MM=IRED(MM)
H
0
MPL=1
MM=MM
is MM=1?
yes
2
no
M=MM(MM)
is M<0?
yes
no
ME(M)=0
E(M)=0
MPL=MPL+1
MM=IRED(MM)


C
is
NPL=N?
yes
no
NELH=0
I=2
Print
Out
End
NELH=NELH+B(I)
is
NELH>1?
yes
A
no
is
B(I)=0?
yes
no
LAST1=I
I=I+1
is
I=N+1?
no
yes
B


R
LN=LN+1
C
D
HN(LN)=LAST1
PRED(LN)=BEF
P(LN)=B(HN)+F(NPL,B(HN),D(HP,LAST1))
BB(LN)=B(LN)
BB(BEF)=99999
M

BRANCH AND BOUND ALGORITHM FOR THE DELIVERY TRUCK PROBLEM

```
      DIMENSION X(15),Y(15),NE(15),NEE(15),B(4000),BB(4000),
     1NFN(4000),NPRED(4000),NZ(4000),ZZ(4000),PCS(15),
     1NROW(15),NCOL(15),DN15,15),DD(15,15)
C
READ N, THE NUMBER OF DESTINATIONS
C
      READ(5,900) N
  900 FORMAT(I2)
C
READ (X,Y), THE COORDINATES FOR EACH DESTINATION
C
      DO 11 I=1,N
      READ(5,901) X(I),Y(I)
  901 FORMAT(2F5.2)
      WRITE(6,333) I,X(I),Y(I)
  333 FORMAT('   I=',I4,'    X=',F5.2,'    Y=',F5.2)
   11 CONTINUE
      DO 750 I=1,N
      DO 750 J=1,N
      IF(I.EQ.J) GO TO 751
      D(I,J)=((X(I)-X(J))**2+(Y(I)-Y(J))**2)**.5
      DD(I,J)=D(I,J)
      GO TO 750
  751 D(I,J)=99999.
      DD(I,J)=99999.
  750 CONTINUE
      LN=1
      B(1)=0.
      BB(1)=0.0
      MN=1
      NFN(1)=1
      NP=1
      NPRED(1)=1
      DO 10 I=1,N
      NEE(I)=1
   10 NE(I)=1
      NBFN=1
      NBTH=1
      NFN=2
      NPL=1
      NNN=N-1
      WT=2200.
  101 IF(NFN(NBFN).LT.0) GO TO 120
      DMIN=99999.
      DNEXT=99999.
      DO 701 J=2,N
      IF(NE(J).EQ.0) GO TO 701
      IF(D(NP,J).GE.DNEXT) GO TO 701
      IF(D(NP,J).GE.DMIN) GO TO 702
      DNEXT=DMIN
      NEXT=NBTH
      DMIN=D(NP,J)
      NBTH=J
      GO TO 701
  702 DNEXT=D(NP,J)
      NEXT=J
  701 CONTINUE
      LN=LN+1
      INTP=LN/100
      NR=LN-INTP*100
      IF(NR.NE.0) GO TO 760
  759 IF(BB(NFN).LT.9990.) GO TO 760
      NFN=NFN+1
```

```
      GO TO 759
760   CONTINUE
      NEN(LN)=NBTH
      NPREC(LN)=NBEN
      B(LN)=B(MN)+.004*(8000.-NPL*270.-B(MN))*DMIN
      BND=B(LN)
      KK=N-NPL-1
      K=0
703   SMALL=99999.
      DO 704 I=2,NNN
      L=I+1
      DO 704 J=L,N
      IF(NEE(I).EQ.0) GO TO 704
      IF(NEE(J).EQ.0) GO TO 704
      IF(SMALL.LE.D(I,J)) GO TO 704
      SMALL=D(I,J)
      II=I
      JJ=J
704   CONTINUE
      K=K+1
      POS(K)=SMALL
      NROW(K)=II
      NCOL(K)=JJ
      D(II,JJ)=99999.
      IF(K.LT.KK) GO TO 703
      DO 705 I=1,KK
      III=NROW(I)
      JJJ=NCOL(I)
      D(III,JJJ)=DD(III,JJJ)
705   CONTINUE
      K=1
      JX=NPL+1
706   FF=.004*(8000.-JX*270.-BND)*POS(K)
      BND=BND+FF
      K=K+1
      JX=JX+1
      IF(K.LE.KK) GO TO 706
      BB(LN)=BND
707   LN=LN+1
      INTP=LN/100
      NR=LN-INTP*100
      IF(NR.NE.0) GO TO 770
769   IF(BB(NEN).LT.9990.) GO TO 770
      NEN=NEN+1
      GO TO 769
770   CONTINUE
      NEN(LN)=-NBTH
      NPREC(LN)=NBEN
      BND=B(MN)+.004*(8000.-NPL*270.-B(MN))*DNEXT
      K=0
709   SMALL=99999.
      DO 708 I=2,NNN
      L=I+1
      DO 708 J=L,N
      IF(NEE(I).EQ.0) GO TO 708
      IF(NEE(J).EQ.0) GO TO 708
      IF(SMALL.LE.D(I,J)) GO TO 708
      SMALL=D(I,J)
      II=I
      JJ=J
708   CONTINUE
      K=K+1
      POS(K)=SMALL
      NROW(K)=II
      NCOL(K)=JJ
      D(II,JJ)=99999.
      IF(K.LT.KK) GO TO 709
      DO 710 I=1,KK
      III=NROW(I)
      JJJ=NCOL(I)
      D(III,JJJ)=DD(III,JJJ)
710   CONTINUE
```

```
      K=1
      JX=NPL+1
  711 FF=.004*(8000.-JX*270.-BND)*POS(K)
      BND=BND+FF
      K=K+1
      JX=JX+1
      IF(K.LE.KK) GO TO 711
      BB(LN)=BND
      NZ(LN)=NEXT
      ZZ(LN)=DNEXT
      BB(NBFN)=99999.
      GO TO 130
  120 NBTH=NZ(NBFN)
      LN=LN+1
      INTP=LN/100
      NR=LN-INTP*100
      IF(NR.NE.0) GO TO 780
  779 IF(BB(NFN).LT.9990.) GO TO 780
      NFN=NFN+1
      GO TO 779
  780 CONTINUE
      NFN(LN)=NBTH
      NPRED(LN)=NBFN
      B(LN)=B(MN)+.004*(8000.-NPL*270.-B(MN))*ZZ(NBFN)
      BND=B(LN)
      KK=N-NPL-1
      K=0
  712 SMALL=99999.
      DO 713 I=2,NNN
      L=I+1
      DO 713 J=L,N
      IF(NEE(I).EQ.0) GO TO 713
      IF(NEE(J).EQ.0) GO TO 713
      IF(SMALL.LE.D(I,J)) GO TO 713
      SMALL=D(I,J)
      II=I
      JJ=J
  713 CONTINUE
      K=K+1
      POS(K)=SMALL
      NROW(K)=II
      NCOL(K)=JJ
      D(II,JJ)=99999.
      IF(K.LT.KK) GO TO 712
      DO 714 I=1,KK
      III=NROW(I)
      JJJ=NCOL(I)
      D(III,JJJ)=DD(III,JJJ)
  714 CONTINUE
      K=1
      JX=NPL+1
  715 FF=.004*(8000.-JX*270.-BND)*POS(K)
      BND=BND+FF
      K=K+1
      JX=JX+1
      IF(K.LE.KK) GO TO 715
      BB(LN)=BND
      NE(NBTH)=0
      DNEXT=99999.
      DO 716 J=2,N
      IF(NE(J).EQ.0) GO TO 716
      IF(D(NP,J).GE.DNEXT) GO TO 716
      DNEXT=D(NP,J)
      NEXT=J
  716 CONTINUE
      GO TO 707
  130 XMINB=9999.
      DO 131 I=NFN,LN
      IF(BB(I).GE.XMINB) GO TO 131
      XMINB=BB(I)
      NBFN=I
  131 CONTINUE
```

```
      DO 140 I=2,N
      NEE(I)=1
  140 NE(I)=1
      MN=NBFN
  141 NP=NHN(MN)
      IF(NP.GT.0) GO TO 150
      NC=-NP
      NE(NO)=C
      MN=NPRED(MN)
      GO TO 141
  150 NPL=1
      MM=MN
  151 IF(MM.EQ.1) GO TO 160
      K=NHN(MM)
      IF(K.LT.0) GO TO 152
      NEE(K)=C
      NE(K)=0
      NPL=NPL+1
  152 MM=NPRED(MM)
      GO TO 151
  160 IF(NPL.EQ.N) GO TO 170
      NELH=0
      DO 161 I=2,N
      NELH=NELH+NE(I)
      IF(NELH.GT.1) GO TO 101
      IF(NE(I).EQ.0) GO TO 161
      LAST1=I
  161 CONTINUE
      LN=LN+1
      INTP=LN/100
      NR=LN-INTP*100
      IF(NR.NE.0) GO TO 730
  729 IF(BB(NFN).LT.9990) GO TO 730
      NFN=NFN+1
      GO TO 729
  730 CONTINUE
      NFN(LN)=LAST1
      NPRED(LN)=NBFN
      B(LN)=B(MN)+.004*(8000.-NPL*270.-B(MN))*D(NP,LAST1)
      BB(LN)=B(LN)
      BB(NBFN)=99999.
      GO TO 130
  170 WRITE(6,949)
  949 FORMAT('1',' OPTIMAL SEQUENCE, IN REVERSE ORDER',///)
      NX=NBFN
  800 NN=NHN(NBFN)
      WRITE(6,950) NN
  950 FORMAT(I10,/)
      IF(NBFN.EQ.1) GO TO 830
  831 NBFN=NPRED(NBFN)
      IF(NHN(NBFN).GT.0) GO TO 800
      GO TO 831
  830 CONTINUE
      WRITE(6,951) LN
  951 FORMAT(///,' NUMBER OF BOUNDS COMPUTED=',I6)
      WRITE(6,952) B(NX)
  952 FORMAT(F12.2)
      WRITE(6,960) NFN
  960 FORMAT('  FN=',I10)
      STOP
      END
```

LIST OF REFERENCES

1. Balut, S.J., Problems in Investigation Theory, Ph.d. Thesis, Naval Postgraduate School, Monterey, Calif., 1973.

2. Bellmore, M., And Nemhauser, G.L., "The Traveling Salesman Problem: A Survey," Operations Research, v. 16, p. 538-558, 1968.

3. DeHaemer, M.J., A Branch-and-Bound Algorithm for the Solution of Sequence Dependent Routing Problems, Master's Thesis, Naval Postgraduate School, Monterey, Calif., 1970.

4. Hayes, R.L., The Delivery Problem, Carnegie Institute of Technology, Management Science Research Report No. 106, July 1967.

5. Held, M., and Karp, R.M., "A Dynamic Programming Approach to Sequencing Problems," SIAM, v. 10, p. 196-210, 1962.

6. Little, J.D.C., and others, "An Algorithm for the Traveling Salesman Problem," Operations Research, v. 11, p. 972-989, 1963.

7. Operations Evaluation Group Internal Memorandum OEG 656-70, Investigation Theory Survey, Some Problems and their Proposed Solutions, by S.J. Balut, 23 Oct 1970.

8. Saaty, T.L., Optimization in Integers and Related Extremal Problems, McGraw-Hill, 1970.

9. Shapiro, D., Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem, Sc.D. Thesis, Washington University, St. Louis, 1966.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Dept. of Operations Research and Administrative Sciences, Code 55 Naval Postgraduate School Monterey, California 93940	1
4.	Chief of Naval Personnel Pers 11b Department of the Navy Washington, D.C. 20370	1
5.	Dept. of Mathematics Naval Postgraduate School Monterey, California 93940	1
6.	Assoc. Professor G.T. Howard Code 55Hk (thesis advisor) Dept. of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	2
7.	LCDR Stephen J. Balut USN SMC 2574 Naval Postgraduate School Monterey, California 93940	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

REPORT TITLE
A Branch and Bound Algorithm for the Delivery Truck Problem

DESCRIPTIVE NOTES (Type of report and, inclusive dates)
Master's Thesis; June 1973

AUTHOR(S) (First name, middle initial, last name)
Stephen John Balut

REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1973	44	9

a CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b PROJECT NO.	
c	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d	

DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

ABSTRACT

The delivery truck problem is one in which a truck is loaded with m packages, one package to be delivered to each of m destinations. The amount of fuel consumed by the truck is directly dependent upon the current total weight of the truck, which includes both the weight of the packages and the amount of fuel remaining in the tank. The problem is to determine a sequence in which to deliver all m packages which will minimize total fuel consumption. A branch and bound algorithm for obtaining optimal solutions to the delivery truck problem is presented, along with several sample problems with their solutions. A brief report of computational experience is included.

DD FORM 1 NOV 68 1473 (PAGE 1)
S/N 0101-807-6811

Security Classification

A-31408

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Sequence Dependent						
Routing Problem						
Branch and Bound						
Delivery Truck Problem						
Investigation Theory						

Thesis
B207
c.1 Balut
A branch and bound algorithm for the delivery truck problem.

145422